Theoretical Computer Science 404 (2008) 52–60

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

(Mem)brane automata

Erzsébet Csuhaj-Varjú [a,b,*], György Vaszil [a]

[a] Computer and Automation Research Institute, Hungarian Academy of Sciences, Kende u.13-17, 1111 Budapest, Hungary
[b] Department of Algorithms and Their Applications, Faculty of Informatics, Eötvös Loránd University, Pázmány P. sétány 1/c, 1117 Budapest, Hungary

ARTICLE INFO

Keywords:
P systems
Brane calculi
P automata
Computational completeness

ABSTRACT

We introduce the notion of a P automaton with marked membranes, a P_{pp} automaton for short, which is an accepting variant of P systems. The concept is motivated by the theory of P systems, brane calculi, and the traditional concept of automata. In P systems with marked membranes, bio-molecules (proteins) are allowed to move through the membranes and to attach onto or to de-attach from the membranes. The membrane system evolves according to rules which are defined over multisets of proteins and describe the above actions. In addition to these features, the P automaton with marked membranes is able to consume inputs from its environment, i.e. multisets of proteins, which might influence the behaviour of the system. The result of the computation is the set of multiset sequences consumed by the skin membrane, supposing that the P_{pp} automaton started functioning in the initial configuration and entered a final configuration at halting. We show that any recursively enumerable language can be obtained as the language accepted by a P_{pp} automaton modulo a simple computable mapping.

1. Introduction

The theory of P systems is a recent vivid scientific field, on the borderline of bio-computing and theoretical computer science. P systems or membrane systems were introduced by Gheorghe Păun in 1998 (the full version of the first article appeared in [18]) in order to introduce a computational concept which mimics the architecture and the functioning of the living cell.

A P system or a membrane system is a structure of hierarchically embedded membranes, each having a label and enclosing a region containing a multiset of objects and possibly other membranes. The outer-most membrane is called the skin membrane. The objects in the regions correspond to bio-chemical ingredients, the membranes to the membranes of the cell. During the functioning of the P system, the objects in the different regions may change and move across the membranes. The rules of the changes and the communication between the membranes can be defined in various manners, thus making it possible to create and study different variants of P systems, with different motivations. The interested reader can find detailed information on P systems in the book [19] as well as on the rapid development of the area in the works referred at the homepage of P systems, http://ppage.psystems.eu.

Brane calculus describes membranes using another approach, the framework of process algebra [3]. It presents a family of process calculi with dynamic nested membranes. In contrast to P systems theory, in brane calculus the computation takes place on the membrane, not inside it, through active entities tightly coupled to membranes.

The first attempt to bridge the two fields was presented in [4] where four basic operations from brane calculi, *pino*, *exo*, *mate* and *drip* were expressed in terms of P systems. The new variants of P systems have marked membranes, i.e. membranes

* Corresponding author at: Computer and Automation Research Institute, Hungarian Academy of Sciences, Kende u.13-17, 1111 Budapest, Hungary.
E-mail addresses: csuhaj@sztaki.hu (E. Csuhaj-Varjú), vaszil@sztaki.hu (G. Vaszil).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.04.004

attached with multisets of symbols called proteins. The evolution of the system is due to rules defined over the markings and applied to them. In contrast to the generic notion of a P system, the evolution of the membrane system is realized both in the change of the structure of the membrane system and in the (change of the) markings. In [4] it was shown that P systems using the *drip* and *mate* operations (even with at most eleven membranes under any computational step) are computationally complete. The result was improved in [1] by proving that P systems with *mate* and *drip* operations and using at most five membranes during any step of the computation are universal.

A model incorporating other features can be found in [2], where so-called membrane systems with marked membranes were introduced. In these membrane systems, bio-molecules (proteins) are allowed to move through the membranes and to attach onto or to de-attach from the membranes. The P system evolves according to the rules which depend on the proteins attached to the membranes. In [2] the evolution rules of the membranes were motivated by the operation of pinocytosis (*pino*) and the operation of cellular dripping (*drip*). In this paper, among other results, it was shown that these systems are computationally universal. Further interesting models were introduced and studied in [16,17] and [15].

During the functioning of the P systems with marked membranes, the skin membrane (the membrane that delimits the membrane system from the environment) remains intact, i.e., it does not communicate with its environment. However, since a membrane system aims at modelling a cell, it is reasonable to examine its behaviour when it communicates with its environment whilst functioning, i.e. to define a variant of a P system with marked membranes which is able to accept inputs from its surrounding environment.

An analogous concept, called P automaton (more precisely, the so-called one-way P automaton), was introduced in the theory of P systems [8,9], where multisets of objects were allowed to enter the skin region step by step during the computation, thus influencing the behaviour of the P system. The notion, in a generalized form, was then defined and studied in [6,7].

Another variant of the accepting P system is the analysing P system introduced in [12]. As standard P automata, these computational devices were computationally complete as well. A lot of variants of accepting P systems or P automata have been developed and studied; for detailed information the reader is referred to the survey [5], the Ph.D dissertation [21], and the articles referred at the P systems webpage.

This idea can also be formulated for P systems with marked membranes. A P automaton with marked membranes or a P_{pp} automaton is a P system with marked membranes which accepts proteins as inputs from its environment. In this case, the skin membrane is allowed to have non-empty marking (this is not allowed for generative P systems with marked membranes) and using these markings the system imports proteins from outside, from the environment. This means that from time to time (not necessarily at every step of the computation) appropriate proteins from the environment attach to the actual marking of the skin membrane and the P system continues its evolution with the obtained new marking at its skin. A protein attached to the skin is called an input (an input protein) for the P_{pp} automaton. The behaviour of the P automaton with marked membranes is characterized by the language accepted by the system. We define the language accepted by a P_{pp} automaton Π as the set of so-called accepted input protein multiset sequences of Π. Such a sequence is a sequence of multisets of proteins attaching to the skin membrane at the different steps of a computation starting from the initial configuration and ending with an accepting configuration with halting. Unlike generative P systems with marked membranes, when defining the result of the computation, i.e. the language of the P_{pp} automaton, we do not require that the system consists of two membranes (the skin and one inside) when it halts.

Since multisets of arbitrary size can be consumed as inputs by the P_{pp} automaton, in order to describe the behaviour of the systems by a language over a finite alphabet, we need a mapping translating the possible infinite alphabet to a finite one. We show that any recursively enumerable language can be accepted by a P_{pp} automaton, modulo a certain very simple, computable mapping.

The reader can see that P_{pp} automaton borrows features from P systems theory (extended with the features motivated by brane calculi) and automata theory. However, a P_{pp} automaton is more of a special accepting system than a standard automata, since the input sequence is not given for the P_{pp} automaton in advance, but it is determined by the functioning of the system.

Obviously, other variants of accepting P systems with proteins on membranes can also be defined, such as in [20].

2. Basic notions

We assume that the reader is familiar with the basic notions of formal language theory and computability. For further details and unexplained notions we refer to [13,14].

We denote by $\mathbb{N}$ the set of all non-negative integers. Let Σ be an alphabet, let Σ^* be the set of all words over Σ, and let $\Sigma^+ = \Sigma^* - \{\lambda\}$ where λ denotes the empty word.

Let V be a set – the universe – of objects. A multiset is a pair $M = (V, f)$, where $f : V \to \mathbb{N}$ is a mapping which assigns a multiplicity to each object $a \in V$. The support of $M = (V, f)$ is the set $supp(M) = \{a \in V \mid f(a) \geq 1\}$. If $supp(M)$ is a finite set, then M is called a finite multiset. The set of all finite multisets over the set V is denoted by V°, the empty multiset is denoted by ϵ.

We say that $a \in M = (V, f)$ if $a \in supp(M)$. $M_1 = (V, f_1) \subseteq M_2 = (V, f_2)$ if for all $a \in V, f_1(a) \leq f_2(a)$. The union of two multisets is defined as $(M_1 \cup M_2) = (V, f')$ where for all $a \in V, f'(a) = f_1(a) + f_2(a)$, the difference is defined for $M_2 \subseteq M_1$ as $(M_1 - M_2) = (V, f'')$ where $f''(a) = f_1(a) - f_2(a)$ for all $a \in V$. We say that M is empty, if its support is empty, $supp(M) = \emptyset$.

The number of objects in a finite multiset $M = (V, f)$, the cardinality of M, is defined by $card(M) = \sum_{a \in V} f(a)$. For a finite set S, the number of elements of S is also denoted by $card(S)$.

A multiset $M = (V, f)$ over the finite set of objects V can be represented as a string w over the alphabet V with $|w| = card(M)$ and $|w|_a = f(a)$ where $a \in V$ and where $|w|$ and $|w|_a$ denote the length of the string w and the number of occurrences of the symbol a in w, respectively.

Throughout the paper, we shall use the notion of a two-counter machine (see [11,14] for further details). A two-counter machine, as defined in [11], is a 3-tape Turing machine, $M = (\Sigma \cup \{Z, B\}, Q, R, q_0, q_f)$ where Σ is an alphabet (the alphabet of input symbols), Q is a set of states with two distinguished elements, $q_0, q_f \in Q$, and R is a set of transition rules. The state q_0 is called the initial state and q_f is called the final state of M. The machine has a read only input tape and two semi-infinite storage tapes (the counters). The alphabet of the storage tapes consists of two symbols, Z and B (blank), while the alphabet of the input tape is $\Sigma \cup \{B\}$. The transition rules of R are of the form $t = \langle x, q, c_1, c_2, q', e_1, e_2, g\rangle$, where $x \in \Sigma \cup \{B\}$ is the symbol scanned on the input tape in state $q \in Q$ and $c_1, c_2 \in \{Z, B\}$ are the symbols scanned on the storage tapes. M enters into state $q' \in Q$, the counters should be modified according to $e_1, e_2 \in \{-1, 0, +1\}$, that is, the counter is incremented by one $(+1)$, the content of the counter remains unchanged (0), or the counter is decremented by one (-1). The input head moves according to $g \in \{0, +1\}$. If $g = +1$, then the head moves one cell to the right, if $g = 0$, then the head remains in the same position.

Symbol Z appears on the cells initially scanned by the storage tape heads and it never appears on any other cell. An integer i can be stored by moving a storage tape head i cells to the right of Z. A stored number can be incremented or decremented by moving the tape head right or left. The machine is capable of checking whether a stored value is zero or not, by looking at the symbol scanned by the storage tape heads. If the scanned symbol is Z, then the value stored in the corresponding counter is zero.

A word $w \in \Sigma^*$ is accepted by the two-counter machine if starting from the initial configuration (having the input word w on the input tape, being in the initial state, and reading Zs on both of the counter tapes), the two-counter machine enters an accepting configuration, that is, the input head scanned the last non-blank symbol and the machine is in the accepting state.

Any recursively enumerable language can be accepted by a two-counter machine; these machines are just as powerful as the Turing machines [11,14].

In the following, we will consider an alternative, equivalent definition of two-counter machines. By omitting the last element of the transition rules, we say that the rules are of the form $t = \langle x, q, c_1, c_2, q', e_1, e_2\rangle$ where $x \in \Sigma \cup \{\lambda\}$. If $x \neq \lambda$ then the symbol is read and the reading head is moved one cell to the right, or if $x = \lambda$ then the transition does not depend on the symbol scanned on the input tape and no move of the reading head is performed. Furthermore, $c_1, c_2 \in \{Z, B, \lambda\}$ where λ has a similar meaning, namely, if $c_i = \lambda,\ i \in \{1, 2\}$, then the transition can be executed irrespective of the symbols scanned on the ith storage tape.

Without the loss of generality, we might also assume that the rules $\langle x, q, c_1, c_2, q', e_1, e_2\rangle$ of the two-counter machine have the following properties:

- If $c_i \neq \lambda$, then $c_{3-i} = \lambda$ and $e_1 = e_2 = 0$, or
- if $e_i \neq 0$, then $e_{3-i} = 0$ and $c_1 = c_2 = \lambda$.

This means that the 7-tuples of R contain, besides the input and the state symbols (the first, second, and third coordinates) only one element which is different from 0 or λ.

Thus, for any transition $t = \langle x, q, c_1, c_2, q', e_1, e_2\rangle$ of the two-counter machine above (according to the alternative, equivalent definition of the original notion) a series of transitions can be constructed such that after performing these transitions in the order of the sequence we obtain the same effect as applying t.

3. Operations for P systems with marked membranes

In standard P system theory, a P system is a structure of hierarchically embedded membranes, each one having a unique label and enclosing a region containing a multiset of objects and possibly other membranes. The outer-most membrane which is unique is called the skin membrane. The membrane structure can be denoted by a sequence of matching parentheses where the matching pairs have the same label as the membranes they represent. The evolution of the system is realized through changing the contents of the regions together – possibly – with changing the membrane structure.

Brane calculi concern operations involving membranes with embedded proteins, formalizing the theory in the framework of process calculi. The approach of [2] borrows features from both theories, namely, it allows proteins (bio-molecules) to move through the regions of the system and to be attached onto (or to be de-attached from) membranes. The membrane system evolves according to the change of the attached proteins.

In this section we recall the basic notions concerning P systems with marked membranes following the terminology used in [2], with adding one membrane operation, called *mate*, [1,4].

Let V be a finite alphabet, called the set of proteins. (We can also use the term object for an element of V.) Let us represent - as it is customary in P systems theory - a membrane with a pair of square brackets, i.e., [ ]. We associate with each membrane a (finite) multiset u over V, denoted by $[\ ]_u$. Then we say that the membrane is marked with u (and u is called a marking for the

membrane). We note that the empty multiset can also be a marking for a membrane. At any moment in time, the membrane is identified by its label in the membrane structure and it has a marking (the multiset u above). If no confusion arises, when presenting a membrane system, we indicate only the markings and the sequence of matching parentheses representing the membrane structure. The content of the membrane might consist of proteins and other membranes.

Operations defined over membranes in P systems with marked membranes are of two types. There are *protein-membrane rules* over the set of proteins V which define the possible changes in the markings and the change in the membrane structure. Furthermore, there are *protein-movement rules* which describe how proteins are allowed to move from inside the region or from outside the region of the membrane or how proteins are allowed to be attached/de-attached to the marking of the membrane from inside or outside of the membrane. In the following, we do not indicate the labels of the membranes, but only their markings.

Let V be a set of proteins and let $a \in V$, $u, v, x \in V^*$. We define the protein-membrane rules $pino_i$, $pino_e$, $drip$, and $mate$ as follows:

$pino_i : [\ ]_{uav} \to [\ [\ ]_{ux}\ ]_v$,

$pino_e : [\ ]_{uav} \to [\ [\ ]_v\ ]_{ux}$,

(letter "i" refers to "internal" and letter "e" refers to "external")

$drip : [\ ]_{uav} \to [\ ]_{ux}\ [\ ]_v$,

$mate : [\ ]_{ua}\ [\ ]_v \to [\ ]_{uxv}$

(We note that in [4], $v \in V^+$ and $ux \in V^+$ is required, but these conditions can be relaxed.) Operation $pino$ is inspired by the pinocytosis, while the $drip$ operation by the cellular dripping. The $mate$ operation is the reverse of $drip$, it joins two cellular structures.

The above rules can be applied to any membrane which is marked by a multiset of proteins containing the marking of the membrane at the left-hand side of the rule. When applying a rule, all the proteins in the marking of the membrane which are not indicated on the left-hand side of the rule are randomly distributed between the two resulting membranes. For the $mate$ operation, the new marking consists of the proteins indicated at the right-hand side of the rule and all other proteins which are present in the markings of the joined cells. The content of the membranes involved in these operations is transferred into the region of the created external membrane (i.e. membrane $[\ ]_v$ in the case of operation $pino_i$ and membrane $[\ ]_{ux}$ in the case of $pino_e$), and in the case of the $drip$ operation the content of the membrane is moved to the region of membrane $[\ ]_v$. In the case of the $mate$ operation, the contents of the starting membranes are put together in the resulting membrane.

Now we define protein-movement rules. As above, let V be a finite set of proteins, $a \in V$, and $u \in V^*$. Then we define

$attach_i : [a]_u \to [\ ]_{ua}$,

$attach_o : [\ ]_u\ a \to [\ ]_{ua}$,

$de-attach_i : [\ ]_{ua} \to [a]_u$,

$de-attach_o : [\ ]_{ua} \to [\ ]_u\ a$,

(letter "i" refers to "inside" and letter "o" refers to "outside")

$move_o : [a]_u \to [\ ]_u\ a$,

$move_i : [\ ]_u\ a \to [a]_u$.

Rules $attach_i$ and $attach_o$ are for attaching protein a to the corresponding membrane if its marking includes the multiset of proteins u. The rules $move_o$ and $move_i$ move the protein a outside of the current membrane or inside from the outer membrane, respectively.

4. P automata with marked membranes

A P automaton with marked membranes or a P_{pp} automaton is a P system with marked membranes which accepts proteins as inputs from its environment. Thus, from time to time (not necessarily at every step of the computation) appropriate proteins from the environment attach to the actual marking of the skin membrane and the P system continues its evolution with the obtained new marking at its skin.

For P automata with marked membranes, we distinguish accepting configurations - in this case at least one membrane contains one or more copies of certain, designated proteins in its marking - and non-accepting ones, where the previous condition does not hold.

A sequence of input protein multisets, i.e. a sequence of protein multisets which were attached after each other to the skin membrane from the environment during the computation, is called an accepted protein sequence, if after consuming the last element of the sequence, the P_{pp} automaton halts in an accepting configuration. Otherwise, it is called a non-accepted input protein sequence.

The sequences of protein multisets consumed from outside can roughly be interpreted as sequences of interactions between the cell and its environment.

In the following we shall present the necessary formal details. Our notions and notations conform with those of [2].

Definition 1. A P automaton with marked membranes, protein-membrane rules, and protein-movement rules (a P_{pp} automaton, for short) is a construct

$$\Pi = (V, \mu, u_1/v_1, \ldots, u_m/v_m, R, R_s, F),$$

where

- V is a finite non-empty set, called the set of proteins,
- μ is a hierarchical membrane structure with $m \geq 1$ membranes; the membrane which is not contained in any membrane is called the skin membrane;
- u_i, $1 \leq i \leq m$, is a multiset of proteins, the initial marking of membrane i in μ,
- v_i is the multiset of proteins present in the region enclosed by membrane i with marking u_i, at the beginning of the computation (the initial content of the region), $1 \leq i \leq m$,
- R is a finite set of protein-membrane rules and protein-movement rules over the alphabet V,
- $R_s \subseteq R$ is the set of rules that are allowed to be applied for the skin membrane; R_s contains rules of type $attach_i$, $attach_o$, $de-attach_i$, $pino_i$, $pino_e$,
- $F \subseteq V$ is a set of proteins that identify the markings of the accepting configurations.

By convention, we label the skin membrane with 1, thus the marking of the skin membrane is u_1 and the content of the skin region is v_1.

Notice the difference between the labeling and the marking of the membranes in the membrane structure. Each membrane in the P systems has a marking, the same marking can be associated with different membranes. Unlike markings, labels identify the membranes in the membrane structure in a unique manner.

The subset of V, which consists of proteins appearing at the left-hand side of the $attach_o$ rules in R_s as proteins to be attached to the marking of the skin membrane, is called set of *input proteins* for Π.

By a configuration of a P_{pp} automaton we mean an $r+1$-tuple $(\mu', u'_1/v'_1, \ldots, u'_r/v'_r)$, $r \geq 1$, where μ' is a membrane structure with r membranes, u'_i is the marking of the membrane i in μ' and v'_i is the multiset of proteins present in the region enclosed by membrane i in μ', $1 \leq i \leq r$. A configuration $(\mu', u'_1/v'_1, \ldots, u'_r/v'_r)$, $r \geq 1$, is an accepting configuration if there is a marking u'_j, $1 \leq j \leq r$, such that u'_j contains at least one element of F.

A P_{pp} automaton changes its configurations through transitions. A transition of a P_{pp} automaton Π from a configuration to another one is performed by the application of its rules for each membrane, i.e., the maximal parallel application of protein-movement rules, or the application of one protein-membrane rule. If for a membrane both protein-movement rules and protein-membrane rules can be applied, then either the protein-movement rules are applied in the maximal parallel manner, or one of the protein-membrane rules is applied. If two or more protein-membrane rules can be applied to a membrane, then one of them is selected in a non-deterministic manner.

The application of the protein-movement rules to a membrane in a non-deterministic maximally parallel way means that the proteins which mark the membrane and the ones which are in the enclosed region are assigned to the rules in such way that after the assignment of the rules no more protein-movement rules can be assigned to the proteins. Any protein is involved in the application of at most one rule.

If an $attach_o$ rule is applied to the skin membrane, then an arbitrary number of copies (but at least one) of the protein is attached to the skin, since the P_{pp} automaton is supposed to have an environment with infinitely many copies of each protein.

The sequence of configurations obtained in the above manner is a computation. Notice that by the definition of the application of rules of type $attach_o$ to the skin membrane, the input multisets appearing under the computation in a P_{pp} automaton form an infinite (countable) alphabet. (The input multisets can be of arbitrary size). Therefore, we introduce a computable mapping $f : V^\circ \to \Sigma \cup \{\lambda\}$ with $f(x) = \lambda$ if and only if $x = \epsilon$, to describe the behaviour of P_{pp} automata with languages over finite alphabets.

Then we define the *language accepted by Π according to f*, denoted by $L(\Pi, f)$, as the set of the sequences of the f-images of protein multisets which are accepted input protein multiset sequences of computations started from the initial configuration of Π, that is, the f-image of an input sequence of Π belongs to the language accepted by Π if and only if the input sequence is an input sequence of multisets of proteins consumed by a computation starting from the initial configuration and ending with an accepting configuration with halting, moreover, the marking of the skin membrane of Π does not contain any input protein at the halting (and accepting) configuration.

5. Computational completeness

In this section we show that P_{pp} automata are as powerful as Turing machines, i.e., they are able to recognize any recursively enumerable language.

Theorem 1. *For any recursively enumerable language L, where $L \subseteq \Sigma^*$, there exists a P_{pp} automaton $\Pi = (V, \mu, u_1/v_1, \ldots, u_4/v_4, R_s, R, F)$ and $f : V^\circ \to \Sigma \cup \{\lambda\}$ with $f(x) = \lambda$ if and only if $x = \epsilon$, such that $L = L(\Pi, f)$.*

Proof. Let $M = (\Sigma \cup \{Z, B\}, Q, P, q_0, q_f)$ be a two-counter machine with $L = L(M)$. We might assume that M has only one transition starting from the initial state, i.e., there is only one transition of the form $t_0 = \langle x, q_0, c_1, c_2, q', e_1, e_2\rangle \in P$. (If this is not the case, we add a new state q'_0 and a new transition $t'_0 = \langle \lambda, q'_0, \lambda, \lambda, q_0, 0, 0\rangle$ to P.) We also assume that there is a unique final transition $t_f = \langle \lambda, q, \lambda, \lambda, q_f, 0, 0\rangle \in P$ where q_f is a state with no $t = \langle x, q_f, c_1, c_2, q', e_1, e_2\rangle \in P$. (If this is not the case, we add a new final state q'_f, and the transition $t'_f = \langle \lambda, q_f, \lambda, \lambda, q'_f, 0, 0\rangle$.)

We construct a P_{pp} automaton accepting L.

Table 1
Rules simulating the reading of the input $x \in \Sigma \cup \{\lambda\}$ by the transition rule $t = \langle x, q, c_1, c_2, q', e_1, e_2\rangle$

Step	Rule	Type	Configuration
0			$[\,[\,]_{\bar{h}\bar{t}z}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s$
1	$[\,]_{\bar{h}\bar{t}} \to [\,]_{\bar{h}}\,[\,]_{h_1h_2t'}$ $[\,]_{\bar{b}c} \to [\,]_{\bar{b}}\,[\,]_{c_1c_2c_{10}}$	drip drip	$[\,[\,]_{\bar{h}z_1}\,[\,]_{h_1h_2t'z_2}\,[\,]_{\bar{b}}\,[\,]_{c_1c_2c_{10}}\,[\,]_d\,]_s$
2	$[\,]_{h_1h_2t'} \to [\,]_{h_3}\,[\,]_{h_2t'}$ $[\,]_{c_1c_2c_{10}} \to [\,]_{c_2}\,[\,]_{c_3c_{10}}$	drip drip	$[\,[\,]_{\bar{h}z_1}\,[\,]_{h_3z_3}\,[\,]_{h_2t'z_4}\,[\,]_{\bar{b}}\,[\,]_{c_2}\cdots$ $\cdots[\,]_{c_3c_{10}}\,[\,]_d\,]_s$
3	$[\,]_{\bar{h}}\,[\,]_{h_3} \to [\,]_{h_4\bar{h}}$ $[\,]_{h_2t'} \to [\,]_{h_2}\,t'$ $[\,]_{c_2}\,[\,]_{c_3c_{10}} \to [\,]_{c_4c_3c_{10}}$	mate de-attach$_o$ mate	$[\,[\,]_{\bar{h}h_4z_1z_3}\,[\,]_{h_2z_4}\,t'\,[\,]_{\bar{b}}\,[\,]_{c_4c_3c_{10}}\,[\,]_d\,]_s$
4	$[\,]_{h_4\bar{h}}\,[\,]_{h_2} \to [\,]_{h_4h_2}$ $[\,]_{c_4c_3c_{10}} \to [\,]_{c_4}\,[\,]_{c_5c_{10}}$ $[\,t'\,]_s \to [\,]_{st'}$	mate drip attach$_i$	$[\,[\,]_{h_4h_2z}\,[\,]_{\bar{b}}\,[\,]_{c_4}\,[\,]_{c_5c_{10}}\,[\,]_d\,]_{st'}$
5	$[\,]_{h_4h_2} \to [\,]_{h_4}\,[\,]_{h_5}$ $[\,]_{c_4}\,[\,]_{c_5c_{10}} \to [\,]_{c_6c_5c_{10}}$ $[\,]_{t'}x \to [\,]_{t'x}$	drip mate attach$_o$	$[\,[\,]_{h_4z_5}\,[\,]_{h_5z_6}\,[\,]_{\bar{b}}\,[\,]_{c_6c_5c_{10}}\,[\,]_d\,]_{st'x}$
6	$[\,]_{h_4}\,[\,]_{h_5} \to [\,]_{h_5h_6}$ $[\,]_{c_6c_5c_{10}} \to [\,]_{c_6}\,[\,]_{c_7c_{10}}$ $[\,]_{st'x} \to [\,[\,]_{t'}\,]_s$	mate drip pino$_i$	$[\,[\,]_{h_5h_6z}\,[\,]_{\bar{b}}\,[\,]_{c_6}\,[\,]_{c_7c_{10}}\,[\,]_d\,[\,]_{t'}\,]_s$
7	$[\,]_{h_5h_6} \to [\,]_{h_6}\,[\,]_{h_7}$ $[\,]_{c_6}\,[\,]_{c_7c_{10}} \to [\,]_{c_8c_7c_{10}}$ $^*[\,]_{sx} \to [\,x\,]_s$	drip mate de-attach$_i$	$[\,[\,]_{h_6z_7}\,[\,]_{h_7z_8}\,[\,]_{\bar{b}}\,[\,]_{c_8c_7c_{10}}\,[\,]_d\,[\,]_{t'}\,]_s$
8	$[\,]_{h_6}\,[\,]_{h_7} \to [\,]_{h_7h_8}$ $[\,]_{c_8c_7c_{10}} \to [\,]_{c_9}\,[\,]_{c_8c_{10}}$ $^*[\,]_d\,x \to [\,]_{dx}$	mate drip attach$_o$	$[\,[\,]_{h_7h_8z}\,[\,]_{\bar{b}}\,[\,]_{c_9}\,[\,]_{c_8c_{10}}\,[\,]_d\,[\,]_{t'}\,]_s$
9	$[\,]_{h_7h_8}\,[\,]_{t'} \to [\,]_{h_8t'h}$ $[\,]_{c_9}\,[\,]_{\bar{b}} \to [\,]_{bc_9}$ $[\,]_{c_8c_{10}} \to [\,]_c\,[\,]_{c_{10}}$ $^*[\,]_{dy} \to [\,]_d\,[\,]_d$	mate mate drip drip	$[\,[\,]_{h_8t'hz}\,[\,]_{bc_9}\,[\,]_c\,[\,]_{c_{10}}\,[\,]_d\,]_s$
10	$[\,]_{h_8t'h} \to [\,]_{h_8h}\,[\,]_t$ $[\,]_{bc_9}\,[\,]_{c_{10}} \to [\,]_{bc_{10}}$ $^*[\,]_d\,[\,]_d \to [\,]_{dd}$	drip mate drip	$[\,[\,]_{h_8hz_9}\,[\,]_{tz_{10}}\,[\,]_{bc_{10}}\,[\,]_c\,[\,]_d\,]_s$
11	$[\,]_{h_8h}\,[\,]_t \to [\,]_{ht}$ $[\,]_{bc_{10}}\,[\,]_c \to [\,]_{bc}$ $^*[\,]_{dd} \to [\,]_d\,[\,]_d$	mate mate drip	$[\,[\,]_{htz}\,[\,]_{bc}\,[\,]_d\,]_s$

The multiset $z \in \{a_1, a_2\}^*$ corresponds to the content of the counters, and furthermore, $z = z_1z_2 = z_1z_3z_4 = z_5z_6 = z_7z_8 = z_9z_{10}$.

Let $\Pi = (V, [\,[\,]_2\,[\,]_3\,[\,]_4\,]_1, s/\lambda, \bar{h}\bar{t}_0/\lambda, \bar{b}c/\lambda, d/\lambda, R, R_s, F)$ where

$$V = \Sigma \cup \{a_1, a_2, h, \bar{h}, b, \bar{b}, c, d, s, r\} \cup \{t, t', t'', \bar{t} \mid t \in P\} \cup \{c_i \mid 1 \le i \le 10\} \cup \{d_i \mid 1 \le i \le 7\} \cup \{h_i \mid 1 \le i \le 8\},$$
$$F = \{t \mid t = \langle x, q, c_1, c_2, q_f, e_1, e_2\rangle \in P\},$$

and the rules of R are constructed as given above. For the ease of reading, we do not list the rules, but present most of them in the form of blocks of rules, given with tables. These tables also demonstrate the effect of the (possible) application of the rules to the given configurations of Π. Those which are not listed in the tables above are presented in the text, with explanations. We note that elements of R_s can also be collected from the corresponding tables.

Each transition of the two-counter machine is simulated in two cycles: the first one enables an appropriate protein to attach to the skin membrane from the environment (if this is not possible then the P_{pp} automaton performs steps to prepare the simulation of the next cycle), and then the manipulation of the counter values are carried out in a second cycle. The value stored in counter i, where $i = 1, 2$ is represented as the number of proteins a_i attached to the membranes of the system. Since the two-counter machine is in the form we presented at the end of Section 2, it is enough to construct groups of rules simulating reading of the input in P, the zero test of a counter, the addition and the subtraction operation. Notice that the check whether or not a counter is non-empty can be simulated by consecutive subtraction and addition instructions.

We note that the constructions used for simulating the zero test, the subtraction, and the addition are based on ideas used in [1], with the necessary modifications.

In the following, for any transition t of M we shall construct a group of rules in R which simulates the application of t.

Let us consider an arbitrary transition $t = \langle x, q, c_1, c_2, q', e_1, e_2\rangle \in P$. The simulation of the reading of protein $x \in \Sigma$ is depicted in Table 1.

Table 2
Rules for the addition of $t = \langle x, q, \lambda, \lambda, q', +1, 0\rangle$ where $z \in \{a_1, a_2\}^*$, and $z' = za_1 = z'_1z'_2 = z'_3z'_4$

Step	Rule	Type	Configuration
0			$[\,[\,]_{htz}\,[\,]_{bc}\,[\,]_d\,]_s$
1	$[\,]_{bc} \to [\,]_{d_1d_2d_3}\,[\,]_c$	drip	$[\,[\,]_{htz}\,[\,]_{d_1d_2d_3}\,[\,]_c\,[\,]_d\,]_s$
2	$[\,]_{d_1d_2d_3} \to [\,]_r\,[\,]_{d_2d_3}$	drip	$[\,[\,]_{htz}\,[\,]_r\,[\,]_{d_2d_3}\,[\,]_c\,[\,]_d\,]_s$
3	$[\,]_{ht}\,[\,]_r \to [\,]_{\bar{h}a_1tr}$ $[\,]_{d_2d_3} \to [\,]_{d_4}\,[\,]_{d_3}$	mate drip	$[\,[\,]_{\bar{h}a_1trz}\,[\,]_{d_4}\,[\,]_{d_3}\,[\,]_c\,[\,]_d\,]_s$
4	$[\,]_{t\bar{h}r} \to [\,]_t\,[\,]_{\bar{h}r}$ $[\,]_{d_4}\,[\,]_{d_3} \to [\,]_{d_5d_6d_3}$	drip mate	$[\,[\,]_{tz'_1}\,[\,]_{\bar{h}rz'_2}\,[\,]_{d_5d_6d_3}\,[\,]_c\,[\,]_d\,]_s$
5	$[\,]_t\,[\,]_{r\bar{h}} \to [\,]_{t''\bar{h}r}$ $[\,]_{d_3d_5d_6} \to [\,]_{d_7}\,[\,]_{d_5d_6}$	mate drip	$[\,[\,]_{\bar{h}t''rz'}\,[\,]_{d_7}\,[\,]_{d_5d_6}\,[\,]_c\,[\,]_d\,]_s$
6	$[\,]_{\bar{h}t''r}\,[\,]_{d_7} \to [\,]_{\bar{h}t''d_7}$ $[\,]_{d_6d_5} \to [\,]_{d_8}\,[\,]_{d_5}$	mate drip	$[\,[\,]_{\bar{h}t''d_7z'}\,[\,]_{d_8}\,[\,]_{d_5}\,[\,]_c\,[\,]_d\,]_s$
7	$[\,]_{\bar{h}t''d_7} \to [\,]_{\bar{h}}\,[\,]_{\bar{u}d_7}$ $[\,]_{d_8}\,[\,]_c \to [\,]_{\bar{b}c}$	drip mate	$[\,[\,]_{\bar{h}z'_3}\,[\,]_{\bar{u}d_7z'_4}\,[\,]_{d_5}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s$
8	$[\,]_{\bar{h}}\,[\,]_{\bar{u}d_7} \to [\,]_{\bar{h}\bar{u}}$ $[\,]_{d_5}\,[\,]_{\bar{b}c} \to [\,]_{\bar{b}c}$	mate mate	$[\,[\,]_{\bar{h}\bar{u}z'}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s$

Starting with a configuration $[\,[\,]_{\bar{h}\bar{t}z}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s$, the P_{pp} automaton reaches in eleven steps the configuration $[\,[\,]_{htz}\,[\,]_{bc}\,[\,]_d\,]_s$ where the multiset $z \in \{a_1, a_2\}^*$ corresponds to the values stored in the two counters. First the symbol t' corresponding to the transition rule $t \in P$ travels to the skin membrane and attaches to it (steps 1–4 in Table 1). Then a symbol x prescribed in the transition t attaches to the skin membrane from the environment (step 5 in Table 1) and then t' moves back inside and returns to form the membrane marking ht, which signals the continuation of the simulation of t (steps 6–8 and steps 9–11 in Table 1). If in step 5, the rule $[\,]_{t'}x \to [\,]_{t'x}$ is used more then once, then the rules marked with * are also used and they produce an infinite cycle by activating rules $[\,]_{dd} \to [\,]_d\,[\,]_d$ and $[\,]_d\,[\,]_d \to [\,]_{dd}$. The computation also enters an infinite cycle if the rules listed in the Table 1 at step i, where $1 \le i \le 11$, are applied in some other order than at the step where they are indicated.

After the eleven steps described above, Π continues its work from the configuration

$$[\,[\,]_{htz}\,[\,]_{bc}\,[\,]_d\,]_s$$

by executing an addition, a zero check (checking whether the content of the counter is zero or not), or a subtraction performed on one of the counters, and then returns to the configuration

$$[\,[\,]_{\bar{h}\bar{u}z'}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s$$

for some $u \in P$ where u is a transition rule which can be applied after the application of t, and z' corresponds to the counter content after the execution of the rule t.

Without any loss of the generality, we shall present the constructions only for the cases when the first counter is tested or modified; the case of the second counter is done in an analogous manner with the obvious modification, namely, by changing a_1 to a_2 in the rules given by the corresponding tables.

For the addition in $t = \langle x, q, \lambda, \lambda, q', +1, 0\rangle$ we need rules as given in Table 2.

Step 1–3 are for introducing the symbol $\bar{h}$ and in the third step one a_1 is added to the marking of the membrane containing symbol t, i.e., the number stored in the first counter is increased by one. Then, in steps 4–8, the P_{pp} automaton enters configuration $[\,[\,]_{\bar{h}\bar{u}z'}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s, z' = za_1$, when the simulation of the addition operation is finished.

Suppose now that $t = \langle x, q, Z, \lambda, q', 0, 0\rangle$, that is, a zero check on the first counter has to be performed. This means that the computation can only continue if no protein a_1 can be found in the marking of membrane 2. The procedure is done by the rules given in Table 3.

The first three steps of the computation produce membranes for maintaining the synchrony, assisting the computation, and introducing symbol $\bar{h}$ appearing in the marking of the second membrane in the configuration $[\,[\,]_{\bar{h}\bar{u}z}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s$ when this phase of the computation is finished. At step 4, the presence of a_1 is tested. If a_1 is present in the counter, then the *drip* rule $[\,]_{ta_1\bar{h}r} \to [\,]_d\,[\,]_{a_1\bar{h}r}$ is performed, leading to an infinite cycle (see rows 10–11 of Table 1). If a_1 is not present, then only the *mate* rule $[\,]_{d_4}\,[\,]_{d_3} \to [\,]_{d_5d_6d_3}$ can be applied. Then, in steps 5–8, the computation leads to a configuration of the form $[\,[\,]_{\bar{h}\bar{u}z}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s$, when this phase of the computation is finished. In addition to the rules in the table, we also need rules $[\,]_{d_5}\,[\,]_{\bar{h}} \to [\,]_{dd\bar{h}}$ and $[\,]_{d_5}\,[\,]_h \to [\,]_{ddh}$, in order to prevent halting of the computation if something goes wrong. Such a situation could arise, for example, if in step 2, the rule $[\,]_{d_2d_3} \to [\,]_{d_4}\,[\,]_{d_3}$ is applied instead of $[\,]_{d_1d_2d_3} \to [\,]_r\,[\,]_{d_2d_3}$, or as another example, the second rule indicated at step 8 is not applied.

The rules for the subtraction, that is, for a transition rule of the form $t = \langle x, q, \lambda, \lambda, q', -1, 0\rangle$ are given in Table 4.

As in the case of the zero test above, the first three steps of this phase of the computation are for introducing the symbol $\bar{h}$, and creating as many membranes as are needed for maintaining the synchrony and assisting the computation. At step 4,

Table 3
Rules for the zero check of $t = \langle x, q, Z, \lambda, q', 0, 0\rangle$ where $z \in \{a_1, a_2\}^*$, and $z = a_1 z_1 z_2$ or $z = z_3 z_4$

Step	Rule	Type	Configuration
0			$[\,[\,]_{htz}\,[\,]_{bc}\,[\,]_{d}\,]_s$
1	$[\,]_{bc} \to [\,]_{d_1 d_2 d_3}\,[\,]_c$	drip	$[\,[\,]_{htz}\,[\,]_{d_1 d_2 d_3}\,[\,]_c\,[\,]_d\,]_s$
2	$[\,]_{d_1 d_2 d_3} \to [\,]_r\,[\,]_{d_2 d_3}$	drip	$[\,[\,]_{htz}\,[\,]_r\,[\,]_{d_2 d_3}\,[\,]_c\,[\,]_d\,]_s$
3	$[\,]_{ht}\,[\,]_r \to [\,]_{\bar{h}tr}$ $[\,]_{d_2 d_3} \to [\,]_{d_4}\,[\,]_{d_3}$	mate drip	$[\,[\,]_{\bar{h}trz}\,[\,]_{d_4}\,[\,]_{d_3}\,[\,]_c\,[\,]_d\,]_s$
4	$[\,]_{ta_1\bar{h}r} \to [\,]_d\,[\,]_{a_1\bar{h}r}$ $[\,]_{d_4}\,[\,]_{d_3} \to [\,]_{d_5 d_6 d_3}$	drip mate	$[\,[\,]_{\bar{h}trz}\,[\,]_{d_5 d_6 d_3}\,[\,]_c\,[\,]_d\,]_s$ or $[\,[\,]_{dz_1}\,[\,]_{a_1 z_2 \bar{h}r}\,[\,]_{d_5 d_6 d_3}\,[\,]_c\,[\,]_d\,]_s$
5	$[\,]_{d_3 d_5 d_6} \to [\,]_{d_7}\,[\,]_{d_5 d_6}$	drip	$[\,[\,]_{\bar{h}trz}\,[\,]_{d_7}\,[\,]_{d_5 d_6}\,[\,]_c\,[\,]_d\,]_s$
6	$[\,]_{\bar{h}tr}\,[\,]_{d_7} \to [\,]_{\bar{h}td_7}$ $[\,]_{d_6 d_5} \to [\,]_{d_8}\,[\,]_{d_5}$	mate drip	$[\,[\,]_{\bar{h}td_7 z}\,[\,]_{d_8}\,[\,]_{d_5}\,[\,]_c\,[\,]_d\,]_s$
7	$[\,]_{\bar{h}td_7} \to [\,]_{\bar{h}}\,[\,]_{\bar{u}d_7}$ $[\,]_{d_8}\,[\,]_c \to [\,]_{\bar{b}c}$	drip mate	$[\,[\,]_{\bar{h}z_3}\,[\,]_{\bar{u}d_7 z_4}\,[\,]_{d_5}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s$
8	$[\,]_{\bar{h}}\,[\,]_{\bar{u}d_7} \to [\,]_{\bar{h}\bar{u}}$ $[\,]_{d_5}\,[\,]_{\bar{b}c} \to [\,]_{\bar{b}c}$	mate mate	$[\,[\,]_{\bar{h}\bar{u}z}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s$

Table 4
Rules for the subtraction of $t = \langle x, q, \lambda, \lambda, q', -1, 0\rangle$ where $z \in \{a_1, a_2\}^*$, and $z = a_1 z_1' z_2' = a_1 z' = a_1 z_3' z_4'$

Step	Rule	Type	Configuration
0			$[\,[\,]_{htz}\,[\,]_{bc}\,[\,]_{d}\,]_s$
1	$[\,]_{bc} \to [\,]_{d_1 d_2 d_3}\,[\,]_c$	drip	$[\,[\,]_{htz}\,[\,]_{d_1 d_2 d_3}\,[\,]_c\,[\,]_d\,]_s$
2	$[\,]_{d_1 d_2 d_3} \to [\,]_r\,[\,]_{d_2 d_3}$	drip	$[\,[\,]_{htz}\,[\,]_r\,[\,]_{d_2 d_3}\,[\,]_c\,[\,]_d\,]_s$
3	$[\,]_{ht}\,[\,]_r \to [\,]_{\bar{h}tr}$ $[\,]_{d_2 d_3} \to [\,]_{d_4}\,[\,]_{d_3}$	mate drip	$[\,[\,]_{\bar{h}trz}\,[\,]_{d_4}\,[\,]_{d_3}\,[\,]_c\,[\,]_d\,]_s$
4	$[\,]_{ta_1\bar{h}r} \to [\,]_t\,[\,]_{\bar{h}r}$ $[\,]_{d_4}\,[\,]_{d_3} \to [\,]_{d_5 d_6 d_3}$	drip mate	$[\,[\,]_{\bar{h}trz}\,[\,]_{d_5 d_6 d_3}\,[\,]_c\,[\,]_d\,]_s$ or $[\,[\,]_{tz_1'}\,[\,]_{\bar{h}rz_2'}\,[\,]_{d_5 d_6 d_3}\,[\,]_c\,[\,]_d\,]_s$
5	$[\,]_t\,[\,]_{r\bar{h}} \to [\,]_{t''\bar{h}r}$ $[\,]_{d_3 d_5 d_6} \to [\,]_{d_7}\,[\,]_{d_5 d_6}$	mate drip	$[\,[\,]_{\bar{h}trz}\,[\,]_{d_7}\,[\,]_{d_5 d_6}\,[\,]_c\,[\,]_d\,]_s$ or $[\,[\,]_{\bar{h}t''rz'}\,[\,]_{d_7}\,[\,]_{d_5 d_6}\,[\,]_c\,[\,]_d\,]_s$
6	$[\,]_{\bar{h}t''r}\,[\,]_{d_7} \to [\,]_{\bar{h}t''d_7}$ $[\,]_{\bar{h}tr}\,[\,]_{d_7} \to [\,]_{\bar{h}trdd}$ $[\,]_{d_6 d_5} \to [\,]_{d_8}\,[\,]_{d_5}$	mate mate drip	$[\,[\,]_{\bar{h}trzdd}\,[\,]_{d_8}\,[\,]_{d_5}\,[\,]_c\,[\,]_d\,]_s$ or $[\,[\,]_{\bar{h}t''z'd_7}\,[\,]_{d_8}\,[\,]_{d_5}\,[\,]_c\,[\,]_d\,]_s$
7	$[\,]_{\bar{h}t''d_7} \to [\,]_{\bar{h}}\,[\,]_{\bar{u}d_7}$ $[\,]_{d_8}\,[\,]_c \to [\,]_{\bar{b}c}$	drip mate	$[\,[\,]_{\bar{h}z_3'}\,[\,]_{\bar{u}d_7 z_4'}\,[\,]_{d_5}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s$
8	$[\,]_{\bar{h}}\,[\,]_{\bar{u}d_7} \to [\,]_{\bar{h}\bar{u}}$ $[\,]_{d_5}\,[\,]_{\bar{b}c} \to [\,]_{\bar{b}c}$	mate mate	$[\,[\,]_{\bar{h}\bar{u}z'}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s$

if symbol a_1 is present in the first counter, it is eliminated by the *drip* rule $[\,]_{ta_1\bar{h}r} \to [\,]_t\,[\,]_{\bar{h}r}$. If there are no symbols a_1 in the first counter, then the computation enters an infinite cycle, due to the application of the rule $[\,]_{d_5}\,[\,]_{\bar{h}} \to [\,]_{dd\bar{h}}$. Steps 5–8 finish this phase of the computation by obtaining a configuration of the form $[\,[\,]_{\bar{h}\bar{u}z'}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s$ where $z'a_1 = z$. The rules $[\,]_{d_5}\,[\,]_h \to [\,]_{ddh}$, $[\,]_{d_5}\,[\,]_{\bar{h}} \to [\,]_{dd\bar{h}}$, and also $[\,]_d\,[\,]_d \to [\,]_{dd}$, prevent halting of the computation if something goes wrong.

When a configuration $[\,[\,]_{\bar{h}t_f}\,[\,]_{\bar{b}c}\,[\,]_d\,]_s$ is obtained, where t_f is the unique final transition of M entering in the final state, q_f, then Π must be able to halt in a few steps, in order to properly simulate the functioning of M. To do this, we add the *mate* rule $[\,]_{\bar{h}t_f}\,[\,]_{\bar{b}c} \to [\,]_{\bar{h}t_f c}$.

Any sequence of transitions accepting a word w in L can be simulated in Π, moreover, if $f : V^\circ \to \Sigma \cup \{\lambda\}$ with

$$f(x) = \begin{cases} a & \text{if } supp(x) = \{a\},\ a \in \Sigma, \text{ or} \\ \epsilon & \text{if } f(x) = \lambda, \end{cases}$$

then the f-image of the accepted input protein sequence of Π is equal to w. Reversely, due to the construction of the rule sets of R, all accepting computations of Π correspond to accepting computations of M, thus the languages accepted by the two constructs are the same. □

It can be shown, by using standard tools, that the language accepted by a P_{pp} automaton Π modulo f is computable by a Turing machine. Therefore, we obtain the following corollary.

Corollary 2. *The class of recursively enumerable languages is equal to the class of languages accepted by P_{pp} automata modulo some f with the properties defined in the proof of Theorem 1.*

6. Conclusions and discussions of the model

In this paper we introduced an accepting computational model, based on P systems with marked membranes, called P_{pp} automata. The construct combined features of P systems and classical automata, and implemented ideas of brane calculi. Although it works by consuming input from its environment, it is different from the standard notion of automata, the input sequence is not given in advance, but it is determined by the functioning of the system. This property makes the construction resemble an abstract model of a functioning living system, a living cell. It would be interesting to study variants of P_{pp} automata where the input sequence is given in advance. According to the construct we presented in the article, the communication is one-way, i.e., the system consumes input but does not return output to the environment. To model interactions between the cell and its environment, formalizing the concept of two-way P_{pp} automata would be useful. Another step to extend, and in some sense to refine the model, would be to consider not a symbol (a protein), but a membrane system as the input (or the input/output). These types of constructions could describe interactions among cell-like systems.

P_{pp} automaton uses both protein-membrane operations and protein-movement rules. In [4] and in [1], however, it was shown that P systems with *mate* and *drip* operations are able to obtain the power of the Turing machines. The question of how to define *P* automata based on purely protein-membrane operations thus naturally arises.

Another interesting topic is the study of the relation between P automata with marked membranes and automata with infinite input alphabets. When consuming the input, the P_{pp} automaton is able to obtain a multiset of arbitrary size in one step. Thus, the inputs form an infinite alphabet. These problems have already been studied, for the case of P automata, in [10].

The comparison of automata theory and the theory of P automata with marked membranes raises other questions as well. For example, while we define the acceptance of the input sequences through both final states and halting, this is not the case for classical automata in general. These and similar problems concerning P_{pp} automata and other accepting variants of P systems with marked membranes are topics for future research, we hope to return to them in the future.

Acknowledgements

This work was supported by the Hungarian Scientific Research Fund “OTKA” Grant no. T042529.

References

[1] D. Besozzi, N. Busi, G. Franco, R. Freund, Gh. Păun, Two universality results for (mem)brane systems, in: M.A. Gutiérrez-Naranjo, A. Riscos-Núñez, F.J. Romero-Campero, D. Sburlan (Eds.), Fourth Brainstorming Week on Membrane Computing, Sevilla, January 30–February 3, 2006, vol. 1, RGNC REPORT 02/2006, Sevilla University, Fénix Editora, Sevilla, 2006, pp. 49–62.
[2] R. Brijder, M. Cavaliere, A. Riscos-Núñez, G. Rozenberg, D. Sburlan, Membrane systems with marked membranes, Electronic Notes in Theoretical Computer Science 171 (2) (2007) 25–36.
[3] L. Cardelli, Brane calculi. Interactions of biological membranes, in: V. Danos, V. Schacter (Eds.), Computational Methods in Systems Biology, CMSB 2004, Paris, France, May 26–28, 2004, in: Lecture Notes in Bioinformatics, vol. 3082, Springer-Verlag, Berlin, 2005, pp. 257–280.
[4] L. Cardelli, Gh. Păun, An universality result for a (mem)brane calculus based on mate/drip operations, International Journal of Foundations of Computer Science 17 (1) (2006) 49–68.
[5] E. Csuhaj-Varjú, P automata, in: G. Mauri, Gh. Păun, M. Pérez-Jimenez, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing: 5th International Workshop, WMC 2004, Milan, Italy, June 2004, in: Lecture Notes in Computer Science, vol. 3365, Springer, 2005, pp. 19–35. Revised Selected and Invited Papers.
[6] E. Csuhaj-Varjú, O.H. Ibarra, Gy. Vaszil, On the computational complexity of P automata, in: C. Ferretti, G. Mauri, C. Zandron (Eds.), DNA 10, Tenth International Meeting on DNA Computing, June 7–10, 2004, University of Milano-Bicocca, Preliminary Proceedings, University of Milano-Bicocca, Milano, 2004, pp. 97–106.
[7] E. Csuhaj-Varjú, O.H. Ibarra, Gy. Vaszil, On the computational complexity of P automata, in: C. Ferretti, G. Mauri, C. Zandron (Eds.), DNA 10, Tenth International Meeting on DNA Computing, June 7–10, 2004, Milan, Italy, in: Lecture Notes in Computer Science, vol. 3384, Springer, 2005, pp. 77–90. Selected papers.
[8] E. Csuhaj-Varjú, Gy. Vaszil, P automata, in: Gh. Păun, C. Zandron (Eds.), Pre-Proceedings of the Workshop on Membrane Computing WMC-CdeA 2002, Curtea de Argeş, Romania, August 19-23, 2002, Pub. No. 1 of MolCoNet-IST-2001-32008, 2002, pp. 177–192.
[9] E. Csuhaj-Varjú, Gy. Vaszil, P automata, in: Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron (Eds.), Membrane Computing, in: Lecture Notes in Computer Science, vol. 2597, Springer, Berlin, 2003, pp. 219–233.
[10] J. Dassow, Gy. Vaszil, P finite automata and regular languages over countably infinite alphabets, in: H.J. Hoogeboom, et al. (Eds.), WMC7, in: Lecture Notes in Computer Science, vol. 4361, 2006, pp. 367–381.
[11] P.C. Fischer, Turing machines with restricted memory access, Information and Control 9 (1966) 364–379.
[12] R. Freund, M. Oswald, A short note on analysing P systems, Bulletin of the EATCS 78 (2002) 231–236.
[13] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vols. I–III, Springer, 1997.
[14] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley, Reading, Mass, 1979.
[15] S.N. Krishna, Combining Brane Calculus and Membrane Computing, in: Proc. of Bio-Inspired Computing - Theory and Applications Conference, Wuhan, China, September 2006, Membrane Computing Section, Bic-TA 2006, pp. 131–143.
[16] A. Păun, B. Popa, P systems with proteins on membranes, Fundamenta Informaticae 72 (4) (2006) 467–483.
[17] A. Păun, B. Popa, P systems with proteins on membranes and membrane division, in: O.H. Ibarra, Z. Dang (Eds.), Proc. of the 10th International Conference on Developments in Language Theory, Santa Barbara, CA, USA, 2006, June 26–29, in: Lecture Notes in Computer Science, vol. 4036, Springer, 2006, pp. 292–303.
[18] Gh. Păun, Computing with membranes, Journal of Computer and System Sciences 61 (1) (2000) 108–143.
[19] Gh. Păun, Membrane computing. An introduction, Springer Verlag, Berlin-Heidelberg, 2002.
[20] B. Popa, Membrane systems with limited parallelism, Ph.D. Dissertation, College of Engineering and Science, Louisiana Technical University, 2006.
[21] M. Oswald, P Automata, Ph.D. Dissertation, Technical University of Vienna, 2003.